# Network Security

# Homework 3

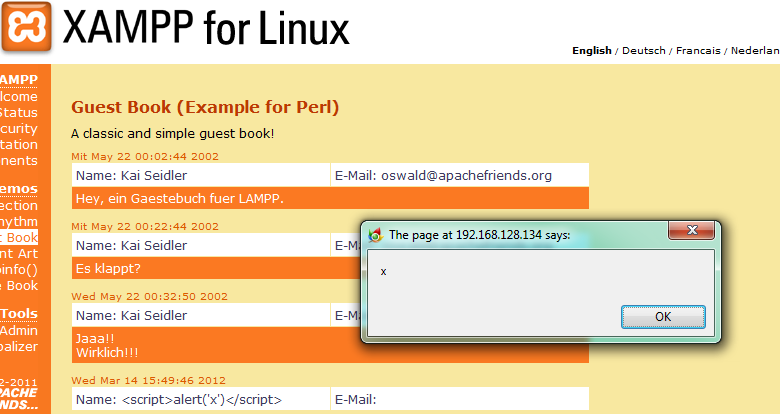
## CHANG LIU

## chang.liu@jhu.edu

# Mar. 14, 2012

# Practical Web Security

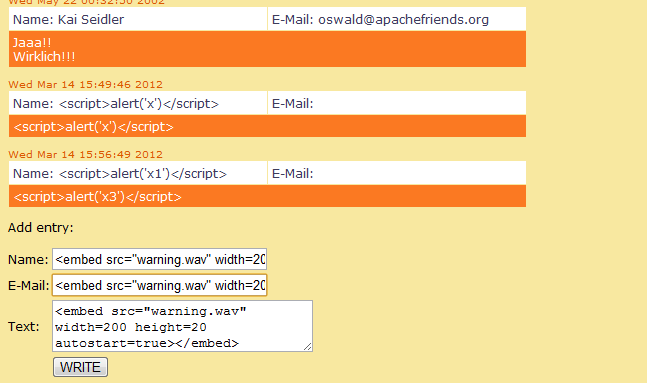
### Task 1



The vulnerability in this website is persistent (stored) XSS, it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on “normal” pages returned to other users in the course of regular browsing.

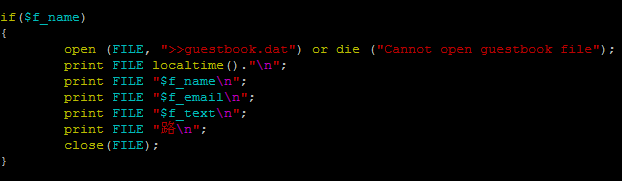
With this security vulnerability, an attacker could store malicious code on the server, and then, when others view the page contains this malicious code, it will automatically executed. For example, when a website stores the user’s username and password in the cookies and this website has XSS vulnerability. An attacker could obtain the users’ username and password information by store the malicious code on the website that will read the users’ cookies and store it somewhere else, and thus get the username and password.

### Task 2



We can use the <embed> tag to make the page play the wave, just input the script as showed in the above figure, it will play the warning.wav automatically once submitted.

### Task 3



From the source code of guestbook-en.pl, we can see that the pl script does NO validation before write the user submitted data into the datafile guestbook.dat. Thus a user could submit malicious code and store it on the server. The solution to fix this problem could be either validate the user’s submitted data and eliminate/transforming any special character “<”, “>” before store it into the database, or, make the content of the stored data non-executable when display the user’s massage from the database.

### Task 4



We can make the SQL injection by provide the query string **‘ or ‘1’ = ‘1**. To mitigate such a vulnerability, the developer could do so by escaping the characters that have special meaning in SQL, such as replace the single quote (‘) with two single quote (‘’).

### Task 5

The HTML 5 local storage allows the programmer to store the browser data locally, such as cookies, flash, etc. It allows much greater amounts of data to be stored locally by the browser, permitting new types of applications. In addition, unlike cookies, the data is not automatically appended to every request by the browser. This is a nice benefit for those attempting to minimize data transmission between the client and server. However, potential vulnerabilities may exist in the local storage: Application may put sensitive data in the local storage like username and password, thus an attacker with physical access to that workstation or that compromises the workstation could get access to that data. Moreover, an attacker could access the data in the local storage remotely with simple XSS script. Thus the local storage of HTML5 may not to be that secure, to mitigate the potential vulnerability, we can apply some effective encryption algorithms to encrypt the stored local data and prevent the unauthorized visitors from accessing the sensitive data.

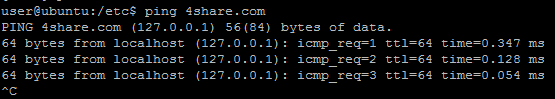
# Practical Network Security

### Task 1

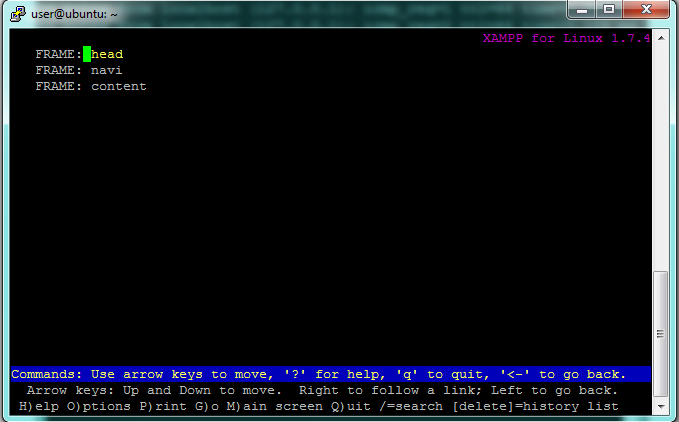
Before we modify the hosts file, we ping 4share.com and see:



After modify the hosts file (added 0.0.0.0 4share.com), we ping the website again and get:



Then visit the 4share.com with the command lynx 4share.com:

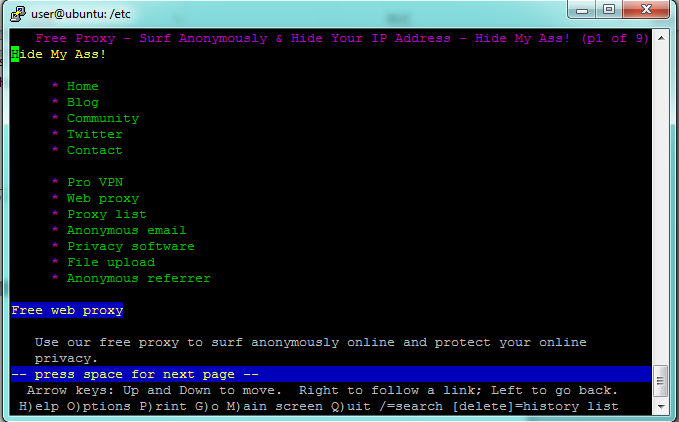


As showed in the two figures above, we get our own IP address (127.0.0.1) when PING the 4share.com, and the 4share.com becomes the XAMPP’s home page. All these occur because we changed the DNS records of 4share.com to 0.0.0.0, which point to the local address of our machine. So when visit 4share.com we actually visit 127.0.0.1, which is the home page of XAMPP.

### Task 2

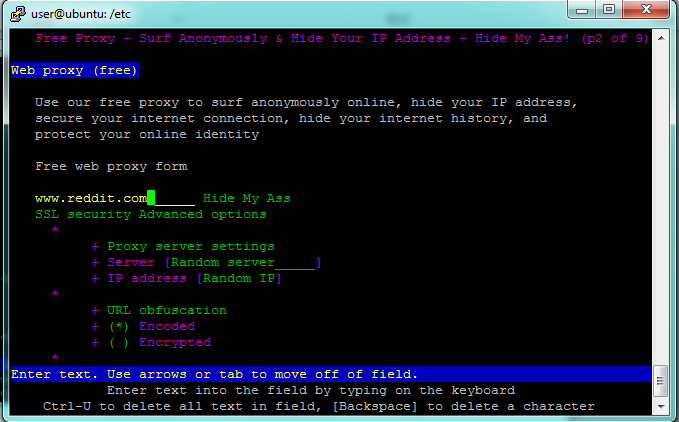
Step 1

lynx to the <http://hidemyass.com/proxy>, as showed in the figure below:



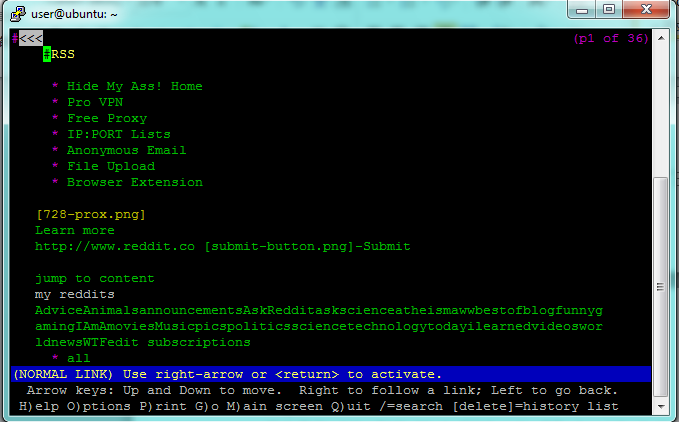
Step 2

Scroll down and enter the URL that the we want to visit, in our case is <http://www.reddit.com>



Step 3

Click (follow) the “Hide My Ass” link, enter “Y” when ask whether allow cookie. Then we will be able to visit the <http://www.reddit.com> via the proxy, as showed in the figure below:



### Task 3

Please see the submitted IP\_submit.sh for details. The following is the script code from that file, I made some modify in the Flushing all rules block so my program reset all iptable rules before execution (that is, it does the same job as fixIP.sh before execution).

#!/bin/sh

# Flushing all rules.

# -F (flush) ; -X (delete policy chain)

iptables -F

iptables -X

iptables -t nat -F

iptables -t nat -X

iptables -t mangle -F

iptables -t mangle -X

iptables -P INPUT ACCEPT

iptables -P FORWARD ACCEPT

iptables -P OUTPUT ACCEPT

# Setting default filter policy

# -P (policy, chain target)

iptables -P INPUT DROP

iptables -P OUTPUT DROP

iptables -P FORWARD DROP

# Allow all established connections

iptables -A INPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A OUTPUT -p TCP -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow ssh

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 22 -j ACCEPT

# Allow ftp

iptables -A INPUT -p tcp --dport 21 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 21 -j ACCEPT

iptables -A INPUT -p tcp --dport 20 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 20 -j ACCEPT

# Allow http

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 80 -j ACCEPT

iptables -A INPUT -p tcp --dport 81 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 81 -j ACCEPT

iptables -A INPUT -p tcp --dport 8080 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 8080 -j ACCEPT

#Allow DNS

iptables -A INPUT -p tcp --dport 53 -j ACCEPT

iptables -A OUTPUT -p tcp --dport 53 -j ACCEPT

# Allow UDP

iptables -A OUTPUT -p udp -j ACCEPT

iptables -A INPUT -p udp -j ACCEPT

# Nothing comes or goes out of this box.

iptables -A INPUT -j DROP

iptables -A OUTPUT -j DROP

### Task 4

I prefer a blacklist approach that you can block all connections, and open ports as needed. This will be more secure than using a whitelist as you only establish the connection that you know/trust, unauthorized connection on the other ports that are not on the list will not be established. Thus it is less likely that a malicious application could establish connection on this blacklist system.

# Proxies

A proxy usually means something channels information or actions, by using something else as the “middle-man” or “go-between” or representative. The main purpose of a proxy is to keep machines behind it anonymous, mainly for security reasons. In our case, the proxy allows the employees access those sites which are prohibited or filtered by our own machine/server. A proxy works in the way that: the requests from the client first reach the proxy where all the requests are treated with filters. And the requests are then passed on to the Hosting server. Then the response from the hosting server again reaches the proxy first, and sends back to the client after filtering on the proxy.

A onion routing is a data structure formed by “wrapping” a plaintext message with successive layers of encryption, such that each layer can be “unwrapped” (decrypted) like the layer of an onion by one intermediary in the succession of intermediaries, with the original plaintext message only being viewable by at most: the sender, the last intermediary and the recipient. If there is end-to-end encryption between the sender and the recipient, then not even the intermediary can view the original message. Thus, when surfing the web using the onion routing, privacy is assured because only the sender and the receiver could see the plaintext. Using a proxy can protect the users’ privacy because proxy makes the machines/users behind it anonymous, and this fact will makes it difficult to trace the Internet activity of a certain user behind the proxy.